# User's Guide

# HONPAS 1.0

June 18, 2023


Xinming Qin *University of Science and Technology of China*

Honghui Shang *University of Science and Technology of China*

Wei Hu *University of Science and Technology of China*

Jinlong Yang *University of Science and Technology of China*


`http://honpas.ustc.edu.cn`

# Contents

# 1 INTRODUCTION

HONPAS ( Hefei Order-N Packages for Ab initio Simulations ) is an ab initio electronic structure calculation software package for linear scaling first-principles density functional theory (DFT) calculations of large-scale systems with standard norm-conserving pseudopotentials and numerical atomic orbitals (NAOs) under the periodic boundary conditions. HONPAS is developed in the framework of the SIESTA methodology and focuses on the development and implementation of efficient linear scaling algorithms for *ab initio* electronic structure calculations.

It provides the following features and functionalities:

- A NAO2GTO scheme to calculate the electron repulsion integrals (ERIs) and their derivatives. Within this scheme, calculations of both total energy and atomic forces with the hybrid functionals (PBE0, B3LYP and HSE06) are available. More accurate calculations for post-HF methods such as Møller–Plesset second-order perturbation (MP2) theory and coupled cluster theory are under development.
- A low rank approximation based on the interpolative separable density fitting (ISDF) decomposition to construct a low rank approximation of HFX matrix, which avoids explicit calculations of ERIs and significantly reduces the computational cost.
- A series of density matrix purification algorithms for the solution of the electronic ground states, including the trace-preserving canonical purification scheme of Palser and Manolopoulos (PM), the trace-correcting purification (TC), and the trace resetting density matrix purification (TRS). The linear-scaling density matrix second-order trace-correcting purification (TC2) algorithm has been extended to perform spin polarized calculations.
- Linear scaling post-SCF calculations for band edge states, doped semiconductors, and the maximally localized Wannier functions (MLWFs).
- Linear scaling method based on the density matrix perturbation theory (DMPT) to treat electric field in solids. Optical dielectric constant and Born effective charges of insulating solids can be calculated with it. Linear scaling phonon calculations based on DMPT will also be provided.

*This Brief Manual only contains descriptions of hybrid functionals available in* HONPAS. For more information you can visit the web page `http://honpas.ustc.edu.cn`.

### References:


- "HONPAS : A linear scaling open-source solution for large system simulations" Xinming Qin, Honghui Shang, Hongjun Xiang, Zhenyu Li and Jinlong Yang, Int. J. Quantum Chem. **115**, 647 (2015)

  Description of the present method and code.

- "Implementation of screened hybrid density functional for periodic systems with numerical atomic orbitals: Basis function fitting and integral screening" Honghui Shang, Zhenyu Li and Jinlong Yang, J. Chem. Phys. **135**, 034110 (2011)

Description of the NAO2GTO method for periodic hybrid functionals.

# 2 COMPILATION

## 2.1 Dependencies

HONPAS (hybrid functional module) requires an external LIBINT library for electron repulsion integrals over Gaussian functions. You can download LIBINT (version 1.1.4 or 1.1.5) from https://sourceforge.net/projects/libint/files/v1-releases/

Follow the following steps to install LIBINT.

```
$ tar -xzvf libint-1.1.5.tar.gz
```

```
$ cd libint-1.1.5
```

```
$ ./configure --prefix=your-libint-install-path CC=icc CXX=icpc
```

Most often you will need to specify command-line options to configure. To obtain a list of configure options run 'configure –help'. "your-libint-install-path" is the specified install directory for LIBINT, and CC/CXX is the C/C++ compiler.

```
$ make -j
```

```
$ make install
```

Remember the install directory of LIBINT, which will be linked by HONPAS

```
LIBINT_DIR=your-libint-install-path
LIBINT_LIBS=${LIBINT_DIR}/lib/libderiv.a ${LIBINT_DIR}/lib/libint.a
```

Also, you can install LIBINT library by using the provided script (`build-libint.sh`) in the top-level `External` of HONPAS.

```
$ cd External
```

```
$ sh build-libint.sh
```

## 2.2 The building directory

The current HONPAS shares the same framework as SIESTA-3.2, including input, output and execution features. In this manual, we assume that you are a proficient user of SIESTA. Otherwise, you should refer to the SIESTA manual at the project's web page http://www.uam.es/siesta.

Rather than using the top-level Src directory as building directory, the user has to use an ad-hoc building directory (by default the top-level `Obj` directory, but it can be any (new) directory in the top level). The building directory will hold the object files, module files, and libraries resulting from the compilation of the sources in `Src`. The `VPATH` mechanism of modern `make` programs is used. This scheme has many advantages. Among them:

- The Src directory is kept pristine.

- Many different object directories can be used concurrently to compile the program with different compilers or optimization levels.

If you just want to compile the program, go to `Obj` and issue the command:

```
sh ../Src/obj_setup.sh
```

to populate this directory with the minimal scaffolding of makefiles, and then make sure that you create or generate an appropriate `arch.make` file (see below, in Sect. 2.3). Then, type

```
make
```

To compile utility programs (those living in `Util`), you can just simply use the provided makefiles, typing "make" as appropriate.

### 2.3 The arch.make file

The compilation of the program is done using a `Makefile` that is provided with the code. This `Makefile` will generate the executable for any of several architectures, with a minimum of tuning required from the user and encapsulated in a separate file called `arch.make`.

You are strongly encouraged to look at `Src/Sys/DOCUMENTED-TEMPLATE.make` for information about the fine points of the `arch.make` file. You can also get inspiration by looking at the actual `arch.make` examples in the `Src/Sys` subdirectory. If you intend to create a parallel version of SIESTA, make sure you have all the extra support libraries (`MPI, scalapack, blacs...` (see Sect. **??**).

Optionally, the command `../Src/configure` will start an automatic scan of your system and try to build an `arch.make` for you. Please note that the configure script might need some help in order to find your Fortran compiler, and that the created `arch.make` may not be optimal, mostly in regard to compiler switches and preprocessor definitions, but the process should provide a reasonable first version. Type `../Src/configure --help` to see the flags understood by the script, and take a look at the `Src/Confs` subdirectory for some examples of their explicit use.

For HONPAS, the `arch.make` file must additionally include a LIBINT dependence:

```
LIBINT_DIR=your-libint-install-path
LIBINT_LIBS=${LIBINT_DIR}/lib/libderiv.a ${LIBINT_DIR}/lib/libint.a
LIBS += $(LIBINT_LIBS) -lstdc++
```

Note that the link order of `libderiv.a` and `libderiv.a` cannot be changed, and `-lstdc++` is needed for FORTRAN to calling C++ library. Check there if there are link errors.

This a simple example of `arch.make`:

```
.SUFFIXES:
```

```
.SUFFIXES: .f .F .o .a .f90 .F90

DUMMY_FOX= --enable-dummy
SIESTA_ARCH=x86_64-unknown-linux-gnu--unknown

FC=mpiifort
RANLIB=ranlib

SYS=nag
SP_KIND=4
DP_KIND=8
KINDS=$(SP_KIND) $(DP_KIND)

FFLAGS= -O2 -g  -assume byterecl -w -fPIC -fp-model source -heap-arrays
FPPFLAGS= -DMPI -DFC_HAVE_FLUSH -DFC_HAVE_ABORT
LDFLAGS= -static-intel

MKL_ROOT=/public/software/intel/2019/mkl/lib/intel64
MKL_LIBS=-L${MKL_ROOT} -lmkl_intel_lp64 -lmkl_core -lmkl_sequential \
               -lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64
LIBS = $(MKL_LIBS) -lpthread

LIBINT_DIR=/public/software/libint/1.1.5/lib
LIBINT_LIBS=${LIBINT_DIR}/lib/libderiv.a ${LIBINT_DIR}/lib/libint.a
LIBS += $(LIBINT_LIBS) -lstdc++

MPI_INTERFACE=libmpi_f90.a
MPI_INCLUDE=.

.F.o:
        $(FC) -c $(FFLAGS) $(INCFLAGS) $(FPPFLAGS) $(FPPFLAGS_fixed_F)  $<
.F90.o:
        $(FC) -c $(FFLAGS) $(INCFLAGS) $(FPPFLAGS) $(FPPFLAGS_free_F90) $<
.f.o:
        $(FC) -c $(FFLAGS) $(INCFLAGS) $(FCFLAGS_fixed_f)  $<
.f90.o:
        $(FC) -c $(FFLAGS) $(INCFLAGS) $(FCFLAGS_free_f90)  $<
```

# 3 EXECUTION OF THE PROGRAM

A fast way to test your installation of HONPAS and get a feeling for the workings of the program is implemented in directory `HONPAS-Examples`. In it you can find several subdirectories with pre-packaged FDF files and pseudopotentials.

We describe here the whole process by means of the simple example of the silicon crystal. It is

advisable to create independent directories for each job, so that everything is clean and neat, and out of the `honpas` directory, so that one can easily update version by replacing the whole `hopas` tree. Go to your favorite working directory and:

```
$ mkdir Sibulk
```

```
$ cd Sibulk
```

```
$ cp path-to-package/HONPAS-Examples/Sibulk/* .
```

You need to make the honpas executable visible in your path. You can do it in many ways, but a simple one is

```
ln -s path-to-package/Obj/honpas .
```

Now you can run the program:

```
./honpas < Sibulk.fdf | tee Sibulk.out
```

(If you are running the parallel version you should use some other invocation, such as

```
mpirun -np 12 honpas ....
```

# 4 THE FLEXIBLE DATA FORMAT (FDF)

The main input file, which is read as the standard input (unit 5), contains all the physical data of the system and the parameters of the simulation to be performed. This file is written in a special format called FDF, developed by Alberto García and José M. Soler. This format allows data to be given in any order, or to be omitted in favor of default values. Refer to documentation in ∼/siesta/Src/fdf for details.

Here we only offer the additional parameters for HSE06 calculations, and all other parameters can be found in the SIESTA manual.

These are all parameters of HSE06 calculations for the Si crystal:

```
XC.functional     GGA
XC.authors        HSE06

%block NAO2GTO
Si   5
3  0  1  5
    0.15909326E+00     0.64313340E+00
    0.86836496E+00     0.19646324E+01
    0.12157109E+01    -0.48790497E+01
    0.17019953E+01     0.28506480E+01
    0.29060337E+01    -0.30906785E+00
3  0  2  5
    0.38851164E+00     0.36778081E+01
    0.54391628E+00    -0.32463429E+01
    0.20857016E+01    -0.22290855E+01
    0.29199823E+01     0.30415158E+01
```

```
    0.40879751E+01    -0.10575846E+01
3  1  1  3
    0.12389670E+00     0.15037097E+00
    0.39288390E+00     0.39394490E+00
    0.72096461E+00    -0.14865807E+00
3  1  2  4
    0.41852167E+00     0.18709405E+01
    0.58593033E+00    -0.17806838E+01
    0.82030244E+00     0.66768749E+00
    0.11484234E+01    -0.14524648E+00
3  2  1  3
    0.37873881E+00     0.44332754E+00
    0.15521261E+01     0.84621145E+00
    0.31042691E+01    -0.16133756E+00
%endblock NAO2GTO

HFX.MinimumNumberGaussians        3
HFX.MaximumNumberGaussians        5

HFX.UseFittedNAOs                 T
HFX.GaussianEPS                0.100E-04

HFX.TruncateDM                    T
HFX.FarField                      T
HFX.FarFieldTolerance     0.100E-05
HFX.PairListTolerance     0.100E-05
HFX.SchwarzTolerance      0.100E-05

HFX.Dynamic_parallel              F
HFX.FragSize               10000
```

The file *fdf.log* contains all the parameters used by HONPAS in a given run, both those specified in the input fdf file and those taken by default. They are written in fdf format, so that you may reuse them as input directly. Input data blocks are copied to the fdf.log file only if you specify the *dump* option for them.

### 4.1 Exchange-correlation functionals

**XC.functional** (*string*): Exchange-correlation functional type. May be `LDA` (local density approximation, equivalent to `LSD`) or `GGA` (Generalized Gradient Approximation).

*Use:* Spin polarization is defined by SpinPolarized label for both `LDA` and `GGA`. There is no difference between `LDA` and `LSD`.

*Value for hybrid functionals:* `GGA`

**XC.authors** (*string*): Particular parametrization of the exchange-correlation functional. Options for HSE06 and PBE0 are:

- `HSE06` (Heyd-Scuseria-Ernzerhof). Hybrid functional approximation. Ref: Heyd, Scuseria and Ernzerhof, JCP 125, 224106 (2006)
- `PBE0` (Hybrid Perdew-Burke-Ernzerhof). Hybrid functional approximation. Ref: Perdew, Burke and Ernzerhof. JCP 110, 5029 (1999); 110, 6158 (1999)

## 4.2 Definition of auxiliary Gaussians

### 4.2.1 The NAO2GTO block

**NAO2GTO** (*data block*): Block with data to define explicitly the auxiliary Gaussian basis to be used. It allows the definition by hand of all the parameters that are used to construct the auxiliary Gaussian basis. The definition of NAO2GTO block is following

```
%block NAO2GTO     # Define NAOGTO fitting
Si   5  # Label, l_shells (double-zeta 3s, double zeta 3p, and polarized 3d )
3  0  1  5  #n, l_shells, Nzeta, number of Gaussians
    0.15909326E+00     0.64313340E+00  # Exponent,  Coefficient
    0.86836496E+00     0.19646324E+01
    0.12157109E+01    -0.48790497E+01
    0.17019953E+01     0.28506480E+01
    0.29060337E+01    -0.30906785E+00
...
%endblock NAO2GTO
```

If this block is not provided manually, an automatic routine will perform the NAO2GTO fitting. Then, the auxiliary Gaussian basis will be generated automatically using the parameters **HFX.MinimumNumberGaussians** and **HFX.MaximumNumberGaussians** , which define the minimum and maximum numbers of Gaussians for NAO2GTO fitting, respectively. In this case, the NAO2GTO block will be output in the main output file, and you can copy them to a new input file since the fitting Gaussians for each species will not change. It is important to stress that, the NAO2GTO block must be regenerated to match new PAOs when the pseudopotential and any NAO-related parameters are changed.

### 4.2.2 Number of auxiliary Gaussians

**HFX.MinimumNumberGaussians** (*integer*): Defines the minimum number of Gaussians for NAO2GTO fitting. It only has an effect when the block **NAOGTO** is not present.

*Default value:* `3`

**HFX.MaximumNumberGaussians** (*integer*): Defines the maximum number of Gaussians for NAO2GTO fitting. It only has an effect when the block **NAOGTO** is not present.

*Default value:* `6`

#### 4.2.3 Replace NAOs with fitted CGTOs

**HFX.UseFittedNAOs** (*logical*): Logical variable to choose the numerical basis for DFT calculations. If this flag is true, fitted orbitals are used as new NAOs for all SIESTA calculation to avoid fitting errors. We feed the values inside a new cutoff radius determined by **HFX.GaussianEPS** back to radial functions, beyond which all values are equal to 0.

*Default value:* `.true.`

**HFX.GaussianEPS** (*real*): Cutoff threshold for fitted contracted orbitals. It offers a general procedure for defining the confining radii of the fitted CGTOs for all the species. The orbital-confining cutoff radius corresponds to the radius where the CGTOs is less than this threshold.

*Default value:* $10^{-4}$

#### 4.2.4 Integral screening

**HFX.TruncatedDM** (*logical*): Logical variable to choose the integral screening method. If this flag is true, the density matrix weighted Schwarz screening will be used for screening electron repulsion integrals.

*Default value:* `.true.`

**HFX.FarField** (*logical*): Logical variable to choose the integral screening method. If this flag is true, the distance screening proposed by Izmaylov et al. will be employed for short-range electron repulsion integrals.

*Default value:* `.true.`

**HFX.SchwarzTolerance** (*real*): Prescreening threshold of ERIs for (density-matrix weighted) Schwarz sceening.

*Default value:* $10^{-6}$

**HFX.PairListTolerance** (*real*): Prescreening threshold of ERIs for building shell orbital pairs.

*Default value:* $10^{-6}$

**HFX.FarFieldTolerance** (*real*): Prescreening threshold of ERIs for far-field sceening.

*Default value:* $10^{-6}$

#### 4.2.5 Parallel options

**HFX.Dynamic_parallel** (*logical*): Logical variable to choose the parallel scheme for building the Hartree-Fock exchange matrix. If enabled, a dynamic parallelization scheme will be used for ERIs. Otherwise, ERIs are simpley parallelized based on a distributions of orbital pairs.

*Default value:* `.false.`

**HFX.FragSize** (*integer*): Specifies the number of batched ERIs for dynamic parallelization. It only has an effect when **HFX.Dynamic_parallel** is true.

*Default value:* 10000

## Index